自然言語処理 73－2
(1989. 6. 29)

例文からの文法獲得に基づく日本文インタフェース構築ツール「ゆい」

"The Japanease Interface Toolkit **YUI**
based on Grammar Acquisition from Japanese Example Sentences. "

杉山高弘 Takahiro SUGIYAMA
吉田宗弘 * Munehiro YOSHIDA
町田和浩 * Kazuhiro MACHIDA
三枝克広 * Katuhiro SAEGUSA
和田孝 Takashi WADA

日本電気 (株) ソフトウェア生産技術開発本部
Software Engineering Development Laboratory, NEC Corporation

* 日本電気技術情報システム開発 (株)
NEC Scientific Information System Development Ltd.

概要 エキスパートシステム、データベース検索や問い合わせシステム等を日本文によって簡単に操作できる日本文インタフェースを構築するツール「ゆい (YUI: Yet another User-Interface)」を開発した。特定分野における限定された日本文を解析するために、今までと全く違う手法を導入する。言語に関する難しい知識や膨大な辞書を必要とせず、既存の応用システムのコマンドを表現する日本文の例文を直接入力することによって文法や語い辞書を生成する。登録された文法と語い辞書を用いるパーザを既存システムに接続することによって日本文インタフェースを構築できる。本構築ツールをエディタ環境構築システムの日本文インタフェースに適用した例をあわせて示す。

ABSTRACT: This paper describes a natural language interface toolkit YUI (Yet another User-Interface) to build Japanese language interfaces for expert systems, data-base inspection systems, and question-answering systems, etc. A newly developed method is proposed to process Japanese sentences depending on a particular domain. Knowledge on natural language processing and large dictionaries is not required for the toolkit's usages. Users input some of the Japanese sentences which represent commands of the application system. From these examples of senteces, the toolkit can acquire a grammar and a word dictionary, appropriate for the application system. The Japanese language interface is built by connecting the application system and a parser with constructed dictionaries. We show an example of a Japanese interface on the platform CANAE for user-interface development using this toolkit.

1 はじめに

近年、計算機の普及によりいままで限られていた人だけが使用していた計算機を広範囲多数の人が利用する機会が著しく増えた。それにともなって計算機になれ親しんでいない人でも、コマンドの代わりに日常の会話文に近い日本語の命令文を入力することによって計算機を容易に使いこなせるようにしたいという要求が高まってきた。これを満たすアプローチの一つとして日本文インタフェースが存在する。

日本文インタフェースが有効に利用できれば、あらゆるアプリケーションシステムが日本文によって実行でき、各システムに依存したコマンドを覚える苦労から解放される。システムは、あいまいや漠然とした人間向きの表現を許した日本語の質問文や命令文を理解し、アプリケーションシステムにふさわしい解析結果を生成する。海外では、データベース検索システムに自然言語インタフェースが接合されたシステム [2] [4] [6] が商品化されている。国内において構文解析システム [7] [9] や小規模な組込型の日本文インタフェースは存在するが、移植性やアプリケーションシステムとの接合を考慮した真に汎用な日本文インタフェースは、以下のような問題点からまだ存在していない。

- 日本文インタフェースを各システムごとに作ることは、通常、自然言語処理に関する難しい知識と膨大な辞書を開発する期間を必要とする。
- 自然言語を扱うシステムに共通して、利用者の質問がシステムで用意されている構文解析文法と辞書で常に解析できるとは限らない。既存のシステムではユーザによって簡単に文法・語い辞書が修正・追加登録できない。

そこで、真に汎用な日本文インタフェース構築ツールの必要性が生じて来る。

本稿で紹介する日本文インタフェース構築ツール「ゆい（YUI: Yet another User-Interface）」は、接合したいアプリケーションシステムに合わせて日本文インタフェースをカスタマイズする。この際、自然言語解析に関する難しい知識や膨大な辞書を作成する開発期間を必要とせず、解析したい日本語例文から登録画面上でその例文を実際に操作することによってどの様な人にでも文法や語いを各辞書に登録できるように支援する。あらかじめ用意されている構文解析部が、登録された辞書を参照しながら入力日本文を構文解析する。このように日本文インタフェース構築ツールが構築した日本文インタフェースは、エンドユーザにアプリケーションシステムを日本語の文章によって利用することを可能にする。

2 構築ツールの基本的アプローチ

特定なアプリケーションシステムを日本文によって操作するインタフェースを開発する際、日本文構文解析部は機械翻訳システムや巨大な構文解析システム等で用いられている大規模な文法・語い辞書やパーザをそのまま流用することは、解析効率・メモリ容量の制限の観点から不可能なことが多い。汎用性や移植性が高い日本文インタフェースとは、膨大な文法と語い辞書を常に備えているシステムではなく、特定な分野に必要十分な文法と語いを誰にでも簡単にカスタマイズできる機能を備えたシステムであると考えた [6]。

この観点に基づいて開発した本構築ツールの特徴を以下に列挙する。

- 例文からの文法生成
  アプリケーション側で必要としているコマンドやプログラムが存在するとき、それを自然言語で言い表した例文を本構築ツールに直接入力することによって、文法を生成していく。日本文インタフェース構築ツールの利用者は難しい言語知識を必要とせず、全く新規にでも文法や語いを辞書に登録できる。例文から文法を生成しているため、余分な文法を登録しない。
- ユニフィケーション文法に基づく構文解析
  構文解析は、ユニフィケーション文法 [10] をベースとしている。登録時に例文から必要十分な文法や語いだけを登録しているため、構文解析は無駄な解析によって実行効率を悪くすることもないし、構文解析結果のあいまい性も少なくなる。
- 文法と生成記述の一体登録
  構文解析文法とその結果から生成・変換するための記述が一体で登録できる。システムの一連の解析動作がローカルに定義されているため理解し易く、保守性も向上する。
- 辞書のコンパクト化
  文法辞書、語い辞書はポータビリティを考慮してユーザが不便を感じない程度に必要最小限の文法と語いを登録することにより日本語インタフェースを構築できる。
- 登録者独自な文節認知単位
  一般ユーザにとって自然言語処理における文法の認識には、だいぶゆらぎがあることが実験的に報告されている [11]。本ツールでは、登録者独自な文節認知単位によって文法を自由に登録できるため、文法登録時にユーザに難しい文法知識を強要することはない。
- アプリケーションシステムとの統合化
  アプリケーションシステムが必要とするコマンドやプログラム言語や記号列等を簡単に生成できる

簡易生成言語を用意する。解析結果をアプリケーションシステムが受理し、解析結果アクセスツールによって必要な部分を取り出し自由に操作もできる。

3 日本文インタフェース構築ツールの構成

本構築ツールは、大きくわけて文法、語い、生成記述を辞書に登録する登録フェーズと、エンドユーザによって入力された日本文を実際に構文解析しアプリケーションシステムにふさわしいコマンドを生成する実行フェーズからなる。

登録フェーズでは、アプリケーションシステム開発者が専用の日本文インタフェースを作成しようとするとき、分野依存の語い・文法辞書の登録・追加・修正をするために用いられる。実行フェーズでは、エンドユーザがアプリケーションシステムを自在に操作するために入力した日本文を、登録部で構築された辞書とあらかじめ用意されている構文解析部を用いて解析し、その解析結果と生成記述からアプリケーションシステムにふさわしい命令文を生成する。

図1で示すように以下の機能から日本文インタフェース構築ツールは構成されている。

1. アプリケーションシステムを操作する日本文を入力する日本語入力部
2. その入力をユニフィケーション文法に基づいて構文解析する構文解析部
3. 構文解析結果を入力としてアプリケーションシステムにふさわしいコマンドや言語を生成する生成記述実行部
4. アプリケーションシステム専用の日本文インタフェースに文法や語い辞書を日本文の例文から登録する辞書登録エディタ
5. 文法・語い辞書、生成記述辞書

4 ユニフィケーション文法に基づく構文解析

本構築ツールが、採用している構文解析手法は、GB理論 [3] やLFG（語い機能文法）[5] で代表されるユニフィケーション文法 [10] に基づく。

ユニフィケーション文法の一例を図2に示す。例示された入力文を解析するためのLFGのC構造規則と構文解析結果のF構造を図の上半分で示している。本ツー

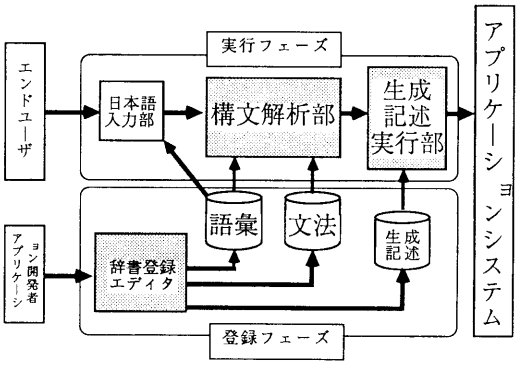

図 1: 日本文インタフェース構築ツールのシステム構成

ルが参照する文法は、内部的にはこれと同等な記述を持っているがユーザに文法を提示する際には、図の下半分で示すように必要十分な情報だけを表示し、そのほかの情報はシステムが自動的に補う。

本ツールの文法は文中のキーワード（述部の語幹が多い）に対して、それとの格の関係（述語項構造）を抜きだしたスケルトンからなる。ここでは、語いや下部構造が、スケルトンのどこに出現するかを明示的に指定するという強い制限を設け、解析の探索範囲を限定することによって解析の効率を向上させる。ただし、自然言語の持つ柔軟性をできる限り失わずに、ユーザに特別な制限をかさずに入力された日本文を解析できる。

本ツールの構文解析手順を以下に簡単に述べる。

1. 入力された解析文を文末からサーチして、キーワードとして登録されている用言の中で最長なものを取り出す。
2. キーワードに対応する文法をすべて取り出す。このときの文法は、ユニフィケーション文法を表すオートマトン（拡張正規表現）をテーブル形式に変換したものである [1]。
3. 解析文と定数項を文字列マッチングさせ、変数項には切り出された語いに対する生成記述内容をバインディングさせる。マッチングとバインディングがすべて成功し、オートマトンが終了状態であれば、構文解析は成功とする。
4. 変数項にバインディングした内容を以下の形式で出力し、生成記述実行部へ渡す。

入力文：　繰越残高と入金額の和を取引後残高に加える

文法(LFG)：
（C構造規則）

S → NP (↑obj=↓, ↓case=ACC) NP (↑obj2=↓, ↓case=DAT) V (↑pred=kuwaeru<(↑obj)(↑obj2)>)

NP → NP (↑adj=↓, ↓case=ADJ or ↑para=↓, ↓case=PARA) N (↑pred=wa<(↑adj)>) P (↑=↓)

NP → N (↑=↓) P (↑=↓)

P → と (↑case=PARA) | の (↑case=ADJ) | を (↑case=ACC) | に (↑case=DAT)

N → 繰越残高 (↑pred=↓) | 入金額 (↑pred=↓) | 和 (↑pred=↓) | 取引後残高 (↑pred=↓)

解析結果：
（F構造）

```
obj
  adj
    para
      case   PARA
      pcase  to
      pred   繰越残高
    case   ADJ
    pcase  no
    pred   入金額
  case   ACC
  pcase  wo
  pred   和(adj)
obj2
  case   DAT
  pcase  ni
  pred   取引後残高
pred     加える((obj)(obj2))
```

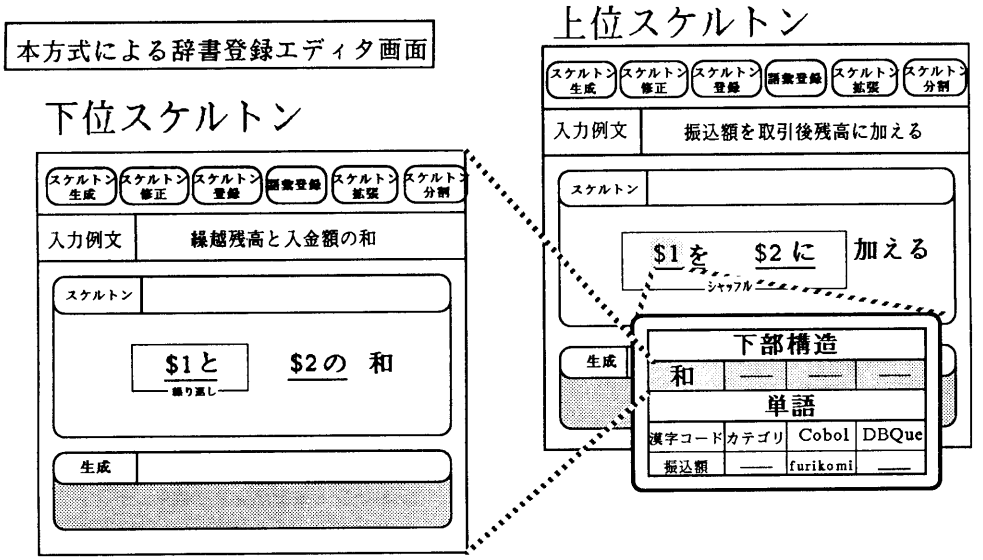

図 2: ユニフィケーション文法

$$binding-list :=$$

$$(skeletonID \left| \begin{array}{l} (\$var\ (content\ \ category)\ldots) \\ (\$var\ binding-list) \end{array} \right| \ldots)$$

図2の入力文を上記手順によって解析すれば、バインディングリストは以下のようになる。図中のLFGのF構造と全く同一構造をしていることがわかる。

```
(<加える> ( $1 (<和> ( $1 (kurikoshi nil ) )
                      ( $2 (nyuukin nil))))
          ( $2 (torihikigo nil)))
```

5 辞書登録エディタ

辞書登録エディタは、難しい言語処理知識を必要とせず例文から直接、文法・語いを作成し、それぞれの辞書に登録する支援をする。これによりユーザフレンドリィなグラフィック画面を用いており、文法登録部、語い登録部、生成記述登録部からなる。この章では、文法登録機能、語い登録機能について詳しく述べる。

5.1 文法辞書登録機能

辞書登録エディタの最も重要な機能である文法辞書登録機能は、構文解析したい日本文の例文を入力することによって文法を登録する。ユニフィケーション文法（LFG等）で採用されているSBJ（主語）、OBJ（目的語）等の構文的に決まる機能を表す**文法機能**や、キーワードとなる述語と文の中に出現する助詞の種類によって決定する**述語項構造**は本ツールが自動的に登録する。

図3で示すように本ツールの文法作成支援として、まず形態素解析ツールを用いて自動的に入力文を語いレベルで分かち書きし、入力文の名詞部分をマークし文節間に区切り記号を挿入する。この解析結果に誤りが生じていたり、登録者独自の文節認知単位に修正したいときは適宜グラフィック画面上で例文を直接操作することによって修正が可能である。ただし、ここでいう文節とは以下の形式に従うものとする。

```
文節     :=   [接頭語]  変数項  [接尾語]  |
              定数項
変数項   :=   <マークされた名詞>
```

マークされた名詞部分を抜き出し必要な部分だけ（主に用言と助詞からなる格構造）を残した文法のひな型を作る。このひな型を**スケルトン**と呼ぶことにする。スケルトンの例を図4に示す。スケルトンは、**定数項**（文字列）と**変数項**からなる。定数項は、スケルトンに残った文字列を表し、構文解析時には解析文と文字列マッチングされる。変数項はスケルトンの抜き出された部分を表し、構文解析時に入力文中の文字列で、かつ、語辞書に登録されている文字列がバインディングされる。登録画面上では$変数に1番から通し番号が自動的に割り当てられる。

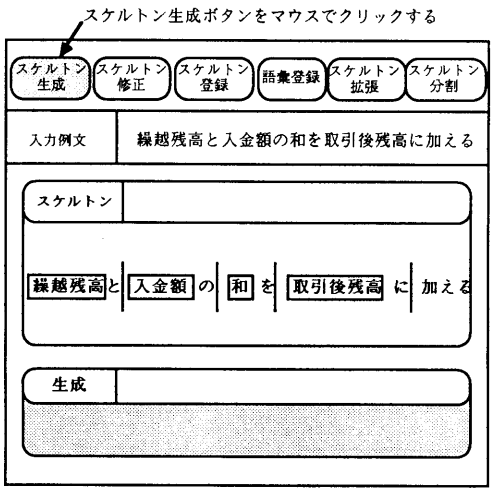

図4: スケルトン生成

最初に入力した例文以外の入力文もスケルトンによって解析できるように、以下の拡張を文法登録エディタは支援する。実際に文法登録エディタでは、図5で示されるように指定する。

- 助詞の出現順の順不同（シャッフル）
  日本文の特徴として用言の格となっている助詞をともなう句は、その出現順を入れ換えても意味がかわらない柔軟な構造を取ることが多い。入力例文上でシャッフル指定することによって、順序の入れ替わった入力文も受理できる。
- 助詞の出現順を固定（シーケンシャル）
  シャッフル指定された内部で助詞の出現順を強制的に固定したいときに指定する。

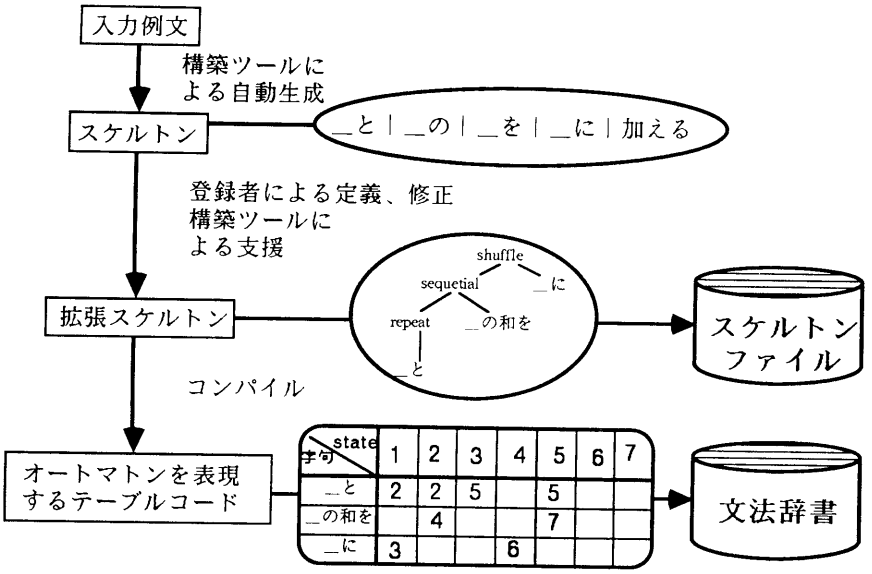

図 3: 文法の作成

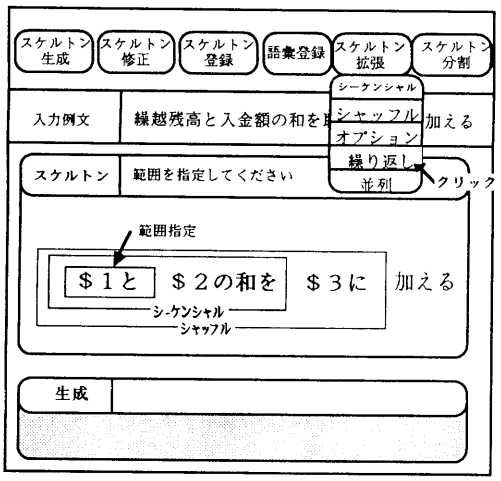

図 5: スケルトン拡張

- 省略可能（オプション）
  オプション指定された句が入力文中に存在していなくても構文解析は成功とする。

- 同一句の繰り返し（繰り返し）
  助詞「と」等の並列句が複数回繰り返されるとき指定する。

- 同等な言い回し（並列）
  どちらの表現でも解析結果に影響がない同等な言い回しを指定する。

- ネスト構造に分割
  生成されたスケルトンにおいてスケルトンの一部分が他のスケルトンに頻繁に出現するとき、その部分をもとのスケルトンから切り出して、ネスト構造の下部スケルトンとする。新たに切り出された下部スケルトンは、ほかのスケルトンの下部構造として再利用が可能となる。これは、図2の登録画面に示されている。

5.2 語い登録機能

構文解析時に参照する語い辞書の登録機能は、アプリケーションシステムへのカスタマイズ機能として重要である。語い登録については、大きく２つの手段がある。１つは、前節で示したスケルトン生成時に入力文中に表れる語いをシステムが自動的に登録する。もう１つは、スケルトンのふさわしい変数項に直接語いを追加登録する。語い辞書に登録する際に、以下で示す３つ組の登録内容とその語いが出現するスケルトンと変数項の位置情報をシステムが自動的に付け加える。

- 入力文中に出現する漢字コード

- 語いをアプリケーション対象分野で用いられる形式に変換した生成オブジェクト
- 生成記述で意味解析を助ける意味カテゴリ

語い登録エディタは、図2の右下にある表入力形式のエディタである。登録したい変数項の位置をマウスで指定することによって登録エディタがポップアップする。

6 生成記述実行部

本生成記述を用いれば、登録者はアプリケーションシステムが必要とするコマンドや構文解析結果を既存のプログラム言語で書くより簡単に記述できる。特に、複雑な条件式や制御文を記述せず、あらかじめ用意された生成記述の機能を用いればよい。また、図6で示すように、記述内容から出力されるイメージが理解しやすい。生成記述と構文解析文法が一体となった登録画面のため、記述がローカルにとじて解析の流れが追いやすく保守性も向上する。現在、Xlib、XToolkit 関数、その上で開発したグラフィック画面操作関数「鼎ライブラリ」[8] 等が利用できるリスプインタプリタによって生成記述は実行される。以下では、生成記述における主な記述を説明する。

1. 生成記述実行
   接合したいアプリケーションによって複数の異なる生成文を出力できるよう、オブジェクト指向的に以下のように記述することができる。
   変数項： メソッド名（選択子）
   変数項には、下部構造としてのスケルトンまたは、語い辞書項目がバインディングされている。スケルトン、語いに関わりなく生成記述を記述することが可能である。スケルトンの生成記述部では、メソッド呼び出しが生じるとメソッド名（選択子）というラベル以下に記述される生成文が実行される。複数の生成文を定義したいときは、1つの登録画面内で複数のメソッド名をアプリケーションに対して定義し、それぞれに生成記述を登録することができる。さらに、同一アプリケーションシステムに関する生成記述でも文脈によって生成文を変えたいという要求がある。具体的には、上位スケルトンの種別によって呼び出される下位スケルトンの生成文を変えるために、メソッド名の引数として選択子を追加することによって文脈に対応した生成が可能である。この例を図6に示す。ただし、選択子をメソッドの引数に記述しないときは、メソッド呼び出し側のスケルトン名がシステムによって与えられる。

2. オプショナル・繰り返し
   オプショナルは、[ ] で指定したバインディング変数に何もバインディングされていなければ、その部分の生成記述を無視する。繰り返しは、( ) * で指定した $変数にバインディングしている各要素について先頭から順にこの生成記述を繰り返す。

3. 外部関数実行
   アプリケーション側が定義した外部関数を日本語インターフェイスの内部からでも呼び出すことができるように、以下のような外部関数実行記述形式を用意する。
   exec: 関数名（引数）

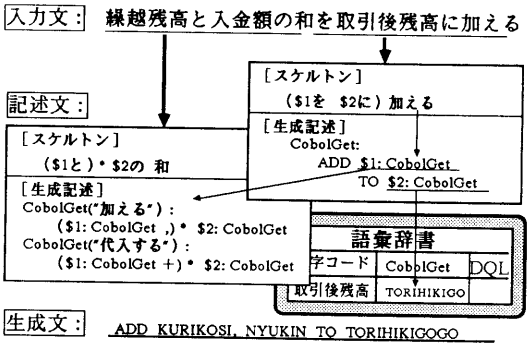

図 6: 生成記述実行例

7 日本文インタフェース例

本構築ツールを用いた例の一つとして、図7に示すようなユーザインタフェース構築環境「鼎」のエディタ部品を日本文の命令文によって操作することによって対話的に生成過程を画面をながめながら設計していくことができるデモシステムを作成した。日本語の文章によりコマンドを高機能化でき、複雑なコマンドを覚えずに抽象的なレベルでシステムを使いこなせることが実証でき、本構築ツールの有効性が確かめられた。

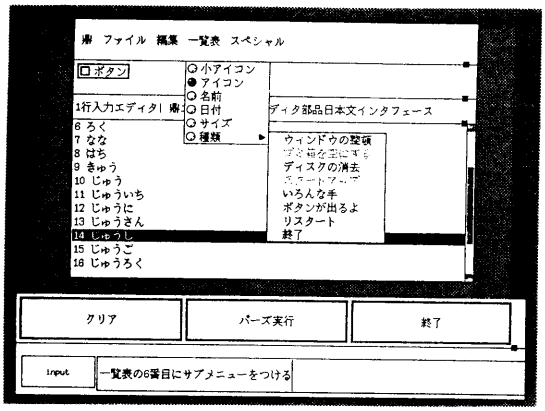

図 7: 鼎エディタ日本文インタフェース

8 まとめ

本稿では汎用な日本文インタフェース構築ツールのために、例文から文法を獲得の辞書登録手段、構文解析方法、アプリケーションシステムへのコマンド生成記述方式をそれぞれ提案した。本ツールを用いることによって難しい自然言語処理知識を必要とせず、例文から文法・語いを獲得することによって日本文インタフェースを構築し、今まで困難とされていた分野のアプリケーションシステムを日本文を用いることによって操作できる。特定なアプリケーションに依存した分野の日本文インタフェースは、エンドユーザにとって必要十分なコンパクトな文法や語い辞書を本ツールの登録エディタから登録することによって移植性を高め、その実現性と有効性が確認できた。

なお、本ツールは SUN3 および、EWS4800 (UNIX SYSTEM V) の X ウィンドウ (X11R2) 上 C 言語で開発した。グラフィック画面作成ツールは「ユーザインタフェース構築環境：鼎（かなえ）」[8] を利用して開発した。

今後の課題として、述部が連用形でつながっている文章や、構造を持った文章等 (1 文中にキーワードの用言がいくつも出てくる文章) の解析、複雑な名詞句解析の本方式への組込等があげられる。

謝辞

研究の機会を与えてくださったソフトウェア生産技術開発本部佐谷本部長、紫合部長、終始有益なご意見を頂いた同川越課長に深謝します。本ツールに関する議論に参加して頂いた基本方式開発部の皆様に感謝します。

参考文献

[1] Alfred V.Aho, Ravi Sethi, Jeffrey D.Ullman, "*Compilers - Principles, Techniques and Tools* -", Addison Wesley (1986).

[2] Barbara J.Grosz *et al.*, "*TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces*", Artificial Intelligence 32(1987) pp.173-243.

[3] Chomsky, N. "*Some Concepts and Consequences of the Theory of Government and Binding*", MIT Press 1982.

[4] Gary G.Hendrix and Brett A.Walter, "The Intelligent Assistant: Q&A", BYTE December 1987 pp.251-258.

[5] Kaplan, R. and J. Bresnan, "*Lexical-Functional Grammar*", MIT Press 1982.

[6] S.Epstein, "*Transportable Natural Language Proccesing through Simplicity - The PRE system*", ACM Trans. on Office Information Systems, Vol.3, No.2, April, 1985.

[7] T.Tanaka, and T.Tokunaga, *et al.*, LangLAB: A Natural Language Analysis System. In COLING88, 1988.

[8] 暦本、菅井、杉山、他「X ウィンドウ上のマルチメディアユーザインタフェース構築環境: 鼎」情報処理学会第 30 回プログラミングシンポジウム予稿集、(1989).

[9] 杉村他「汎用日本語処理系 LTB の構成」他情報処理学会第 37 回全国大会、1988、PP.1072-1089.

[10] 淵監修「自然言語の基礎理論」共立出版、(1986).

[11] 原田悦子「日本語テキストにおける認知的単位」情報処理学会、SIGDPHI 22-3 (1989).